BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

(19)日本国特許庁(JP)　(12)公開特許公報（A）　(11)特許出願公開番号

特開平８－２９２８８７

(43)公開日　平成8年(1996)11月5日

(51)Int.Cl.[6]	識別記号	庁内整理番号	FI		技術表示箇所
G06F	9/38	330	G06F	9/38 330 C	

審査請求　未請求　請求項の数６　FD　(全１２頁)

(21)出願番号　特願平8-58296

(22)出願日　平成8年(1996)2月21日

(31)優先権主張番号　特願平7-62198

(32)優先日　平7(1995)2月24日

(33)優先権主張国　日本（ＪＰ）

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目6番地

(72)発明者　島田　健太郎
東京都国分寺市東恋ケ窪1丁目280番地　株式会社日立製作所中央研究所内

(72)発明者　花輪　誠
東京都国分寺市東恋ケ窪1丁目280番地　株式会社日立製作所中央研究所内

(72)発明者　山本　一道
東京都国分寺市東恋ケ窪1丁目280番地　株式会社日立製作所中央研究所内

(74)代理人　弁理士　笹岡　茂　(外1名)

最終頁に続く

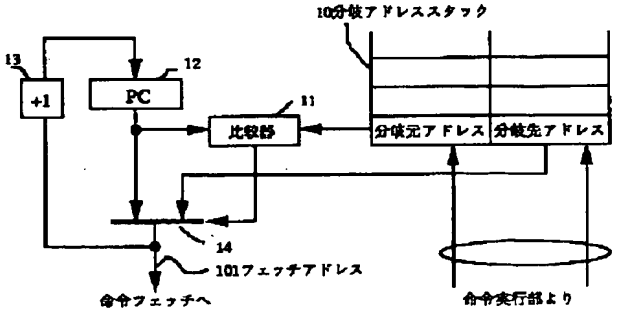
(54)【発明の名称】命令の分岐方法およびプロセッサ

(57)【要約】

【課題】　プログラムのコンパイル時に生成される分岐予約命令を用いて、パイプライン化されたプロセッサでの命令の分岐動作を効率化することにある。

【解決手段】　分岐元アドレスと分岐先アドレスの複数の組を保持する分岐アドレススタック（以下、スタック）10と、現在の命令フェッチアドレスを保持するプログラムカウンタ（ＰＣ）12と、スタック10の保持する最新の組の内の分岐元アドレスとＰＣ12の値とを比較する比較器11と、比較の結果が一致のときに命令フェッチアドレスをＰＣ12の値からスタック10の保持する最新の組の内の分岐先アドレスに切り替えるセレクタ14を備えたプロセッサで、命令列中の分岐点に先立って、分岐予約命令によりスタック10に分岐先アドレスと分岐元アドレスの組を予約しておき、命令の読み込みが分岐点に達したならば速やかに次の命令読み込みアドレスを分岐先に切り替えることで、分岐動作を行う。

【図１】



1

【特許請求の範囲】

【請求項１】　プロセッサにおける命令の分岐方法であって、

分岐予約命令の実行に応じて該分岐予約命令が指定する分岐元のアドレスと分岐先のアドレスを予約するステップと、

命令の読み込みを行うアドレスが前記予約した分岐元のアドレスに達したか否か比較判定するステップと、

前記比較判定の結果、前記命令の読み込みを行うアドレスが前記予約した分岐元のアドレスに達したとき、該命令の読み込みを行うアドレスを前記予約した分岐先のアドレスに切り替えるステップからなることを特徴とする命令の分岐方法。

【請求項２】　請求項１記載の命令の分岐方法において、

前記分岐予約命令が複数個順次実行されたとき、前記分岐予約命令が指定する分岐元のアドレスと分岐先のアドレスを予約するステップを複数回繰り返すことにより複数の分岐を予約し、

前記比較判定するステップにおいては、前記予約された複数の分岐の内最も最近に予約された分岐元のアドレスを選択して前記比較判定をすることを特徴とする命令の分岐方法。

【請求項３】　請求項１または請求項２記載の命令の分岐方法において、

分岐予約命令の実行により分岐元のアドレスと分岐先のアドレスの予約が行われた後で分岐が行なわれる前記分岐元のアドレスに達する前に前記分岐元のアドレスと分岐先のアドレスの予約を条件的に取り消す命令が配置されている場合、該条件的に取り消す命令の実行に応じて前記予約した分岐元のアドレスと分岐先のアドレスを条件的に取り消すステップを設けることにより命令の条件付き分岐を行なうことを特徴とする命令の分岐方法。

【請求項４】　命令読み込み手段と、少なくとも一つの命令解読手段と、少なくとも一つの命令実行手段を備え、前記命令読み込み手段と命令解読手段と命令実行手段が並列に動作可能なプロセッサにおいて、

前記命令読み込み手段は、命令の読み込みを行うアドレスを保持するプログラム・カウンタと、

分岐予約命令が実行されたとき該分岐予約命令が指定する分岐元のアドレスと分岐先のアドレスの組を格納保持する分岐アドレス組保持手段と、

前記プログラム・カウンタの保持する値と前記分岐アドレス組保持手段の保持する分岐アドレス組の内の分岐元のアドレスとを比較する比較手段と、

前記比較手段による比較の結果が一致を示すとき、前記プログラム・カウンタの保持する値を前記分岐アドレス組保持手段の保持する分岐アドレス組の内の分岐先アドレスの値に切り替えて出力する切り替え手段を備え、

命令の読み込みを行うアドレスが分岐予約命令の指定した分岐元のアドレスと一致したとき、分岐予約命令が指定した分岐先のアドレスの命令へ分岐を行うことを特徴とするプロセッサ。

2

【請求項５】　請求項４記載のプロセッサにおいて、

前記分岐アドレス組保持手段は、分岐先アドレスと分岐元アドレスの組を複数保持でき、

前記比較手段は、前記分岐アドレス組保持手段に保持された分岐アドレス組の内の最新の組を選択し、該選択した組の分岐元のアドレスの値と前記プログラム・カウンタの保持する値とを比較することを特徴とするプロセッサ。

【請求項６】　請求項４または請求項５記載のプロセッサにおいて、

前記分岐アドレス組保持手段は、前記命令実行手段からの取り消し信号を受けて該取り消し信号の指定する分岐元のアドレスと分岐先のアドレスの組を消去する手段を備えることを特徴とするプロセッサ。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、プロセッサに関し、特に命令の読み出しと命令の実行がパイプライン化されて並列に行われるプロセッサに適する命令の分岐方法とそのための装置に関するものである。

【０００２】

【従来の技術】まず、従来のパイプライン化されたプロセッサにおける分岐動作について述べる。なお、ここで述べる従来のパイプライン化されたプロセッサにおける分岐動作については、文献「Ｃｏｍｐｕｔｅｒ　Ａｒｃｈｉｔｅｃｔｕｒｅ：Ａ　Ｑｕａｎｔａｔｉｖｅ　Ａｐｐｒｏａｃｈ」（Ｄａｖｉｄ　Ａ．Ｐａｔｔｅｒｓｏｎ　ａｎｄ　Ｊｏｈｎ　Ｌ．Ｈｅｎｎｎｅｓｓｙ、Ｍｏｒｇａｎ　ＫａｕｆｍａｎｎＰｕｂｌｉｓｈｅｒｓ、Ｉｎｃ．、１９９０）で、特に６章においてよくまとめられている。一般的なパイプライン化されたプロセッサでは、高速化のために、命令の実行とこれに後続する命令の読み込みが並列に行われている。しかし分岐命令が実行されてかつ分岐が生じる時には、分岐命令に後続する命令は実行されないので、その読み込みは無駄になる。更に従来では、分岐命令の実行の結果初めて分岐先が定まるので、分岐先の命令の読み込みが行えるようになるまでには時間がかかる。パイプライン化されたプロセッサでは、このように分岐命令においてパイプラインによる命令の読み込みと実行の並列動作が不可となるので、大きく性能を落とす要因となっていた。

【０００３】そこで従来では、分岐命令に続く後続命令を１命令だけそのまま実行することによって、後続命令の読み込みを有効に利用しようとする方策が採られていた。これは、見かけ上分岐が生じるのが分岐命令の次の命令の後に遅れるように見えることから、遅延分岐方式と呼ばれる。プロセッサの回路方式を工夫すれば、更に

分岐先アドレスの計算も後続命令を読み込んでいる間に行うことが可能であり、そのように構成すれば、後続命令の次に続けて分岐先の命令の読み込みを行うことも出来る。このようにすることによってパイプライン化されたプロセッサにおいても分岐動作を効率良く行おうとしていた。

【０００４】ところが、オペレーティング・システムなどプロセスの管理を行うようなプログラムでは、分岐先アドレスの計算をレジスタ等の値を用いて動的に行う分岐命令が用いられている。しかし、このような複雑な分岐先アドレスの計算を行うと、後続命令をたかだか１命令読み込んでいる時間では計算が終了しないと言うことも生じる。また分岐命令につづいて実行される後続命令１命令は、特にその分岐命令が条件分岐命令の時は分岐の生起／不生起に関わらず実行できることが条件となる。しかし、そのような命令を常に見付けることは、プログラムの性質にもより、完全にはできないことが多い。

【０００５】さらに、近年では高性能化のために、一つのプロセッサ内に複数の命令実行パイプラインを用意し、複数の命令を同時に読み込んで並列に実行させる、スーパースカラ方式やＶＬＩＷ（Ｖｅｒｙ　Ｌｏｎｇ　Ｉｎｓｔｒｕｃｔｉｏｎ　Ｗｏｒｄ）方式と呼ばれる命令単位の並列実行方式を採るプロセッサも多くなって来ている。このようなプロセッサでは、分岐命令とそれに後続する命令がしばしば同時に読み込まれてしまうので、遅延分岐方式のように後続命令の読み込み時間で分岐命令の実行時間を隠すことができない。結果として、分岐命令（を含む複数命令）の読み込みから次に実行すべき分岐先の命令を読み込めるようになるまでの時間が顕著に現われてしまい、このような命令単位の並列実行方式を採るプロセッサにおける大きな問題となっていた。

【０００６】このような問題に対する従来の解決策は、分岐予測方式である。これには、静的な分岐予測と動的な分岐予測の二つの基本方式がある。静的な分岐予測では、分岐命令のタイプと分岐先が前方か後方かなどの静的な情報によって分岐が起こるかどうかを予測する。動的な分岐予測では、その分岐命令で過去に分岐したかどうかの履歴をある程度記録しておき、その履歴に基づいて一定のアルゴリズムで次に分岐が起こるかどうかを予測する。プロセッサ内の命令の読み込み部では、読み込んだ（複数）命令中に分岐命令があるかどうかを検出する。分岐命令があった場合には、分岐予測の結果に従って次に読み込む命令を後続の命令にするか分岐先の命令にするかを決定する。この命令読み込み部における分岐命令の検出は、すぐ次の命令読み込みに間に合うように充分素早く行う必要がある。従来の分岐予測方式では、分岐命令があったアドレスをその分岐先アドレスと組にして記録しておく、内容（の一部）での検索が可能な連想メモリを用いたテーブルを用意するという方法を採っている。即ち、それぞれの分岐命令の初回の実行でその分岐命令が置かれていたアドレスをその実行の結果得られる分岐先アドレスと共に記録しておき、新しく命令を読む込む度にそのアドレスが記録されていないかどうかそのテーブルを検索する。このようにすることによって、それぞれの分岐命令が２回目以降、早期に検出でき、分岐先のアドレスも得ることができる。またテーブルには分岐命令の置かれているアドレス及び分岐先のアドレスと組にして、その分岐命令の過去の分岐発生の履歴に関する情報を記録しておけば、分岐予測に関する機構も同時に実装することが可能である。

【０００７】このように従来では分岐予測方式によって、パイプライン化されたプロセッサで分岐をさらに効率化しようとしていた。しかし、従来では分岐命令の初めの実行によってその分岐命令の置かれているアドレスと分岐先のアドレスを登録するので、初回の実行は常に予測ができない。また登録するテーブルも限られた大きさであり、多数の分岐命令が存在するときはＬＲＵ（Ｌｅａｓｔ　Ｒｅｃｅｎｔｌｙ　Ｕｓｅｄ）など一定のアルゴリズムで入れ替えを行わなければならない。入れ替えの結果追い出された分岐命令は、再び予測不可となる。またさらに大きな問題としては、分岐予測の精度がある。即ち、分岐予測方式では、実際の分岐命令の実行に先だってあくまで予測をするだけであるので、予測がはずれることも充分有り得る。このような予測の精度は、先に述べたような分岐予測の方法にもよるが、実行するプログラムの性質にもよる。即ち、実行する度に毎回分岐の生起／不生起が変化するような分岐命令を含むようなプログラムでは、予測の精度を上げることは非常に困難である。予測がはずれた場合には、予測の結果に基づいて読み込んでいた命令を一旦すべて取り消して、元の分岐点まで戻って命令の読み込みからやり直さなければならない。その結果、予測しなかった場合よりもかえって余分な時間がかかる場合も多い。

【０００８】

【発明が解決しようとする課題】本発明は、分岐先の決定は多くの場合、プログラムのコンパイル時に予め静的に、あるいはサブルーチンからの戻り（リターン）などのように動的に決定される場合にも、命令列中で分岐をしなければならない分岐点に充分先立って決定できることに着目し、前述した問題点を解消しようとするものである。本発明の目的は、プログラムのコンパイル時に生成される分岐予約命令によって、パイプライン化されたプロセッサで命令の読み込みと命令の実行が並列に行われていても、効率良く分岐を行う方法およびプロセッサを提供することにある。

【０００９】

【課題を解決するための手段】上記目的を達成するため、本発明は、プロセッサにおける命令の分岐方法であ

り、分岐予約命令の実行に応じて該分岐予約命令が指定する分岐元のアドレスと分岐先のアドレスを予約するステップと、命令の読み込みを行うアドレスが前記予約した分岐元のアドレスに達したか否か比較判定するステップと、前記比較判定の結果、前記命令の読み込みを行うアドレスが前記予約した分岐元のアドレスに達したとき、該命令の読み込みを行うアドレスを前記予約した分岐先のアドレスに切り替えるステップからなるようにしている。さらに、前記分岐予約命令が複数個順次実行されたとき、前記分岐予約命令が指定する分岐元のアドレスと分岐先のアドレスを予約するステップを複数回繰り返すことにより複数の分岐を予約し、前記比較判定するステップにおいては、前記予約された複数の分岐の内最も最近に予約された分岐元のアドレスを選択して前記比較判定をするようにしている。さらに、分岐予約命令の実行により分岐元のアドレスと分岐先のアドレスの予約が行われた後で分岐が行なわれる前記分岐元のアドレスに達する前に前記分岐の予約を条件的に取り消す命令が配置されている場合、該分岐予約命令を条件的に取り消す命令の読み出しに応じて前記予約した分岐元のアドレスと分岐先のアドレスを条件的に取り消すステップを設けることにより命令の条件付き分岐を行なうようにしている。また、命令読み込み手段と、少なくとも一つの命令解読手段と、少なくとも一つの命令実行手段を備え、前記命令読み込み手段と命令解読手段と命令実行手段が並列に動作可能なプロセッサにおいて、前記命令読み込み手段は、命令の読み込みを行うアドレスを保持するプログラム・カウンタと、分岐予約命令が実行されたとき該分岐予約命令が指定する分岐元のアドレスと分岐先のアドレスの組を格納保持する分岐アドレス組保持手段と、前記プログラム・カウンタの保持する値と前記分岐アドレス組保持手段の保持する分岐アドレス組の内の分岐元のアドレスとを比較する比較手段と、前記比較手段による比較の結果が一致を示すとき、前記プログラム・カウンタの保持する値を前記分岐アドレス組保持手段の保持する分岐アドレス組の内の分岐先アドレスの値に切り替えて出力する切り替え手段を備え、命令の読み込みを行うアドレスが分岐予約命令の指定した分岐元のアドレスと一致したとき、分岐予約命令が指定した分岐先のアドレスの命令へ分岐を行うようにしている。さらに、前記分岐アドレス組保持手段は、分岐先アドレスと分岐元アドレスの組を複数保持でき、前記比較手段は、前記分岐アドレス組保持手段に保持された分岐アドレス組の内の最新の組を選択し、該選択した組の分岐元のアドレスの値と前記プログラム・カウンタの保持する値とを比較するようにしている。さらに、前記分岐アドレス組保持手段は、前記命令実行手段からの取り消し信号を受けて該取り消し信号の指定する分岐元のアドレスと分岐先のアドレスの組を消去する手段を備えるようにしている。

【００１０】

【実施例】本発明の実施例を図によって説明する。図１は、本発明の一実施例を図示したものである。本発明はプロセッサの、特に命令フェッチ（読み込み）アドレス生成部において実施されるので、図１では命令フェッチアドレス生成部に限って示してある。プロセッサ全体の構成については、一例を図２に掲げておく。図１では、命令実行部で分岐予約命令が実行されることにより、分岐元アドレスと分岐先アドレスが分岐アドレススタック１０に登録される。上記分岐予約命令はプログラムのコンパイル時に予め作成され、分岐点に充分先立って命令列中に配置される。すなわち、分岐点における分岐先の決定は、多くの場合、予め「静的」＝「プログラムのコンパイル時」に或いはサブルーチンからの戻り（リターン）などのように動的に決定される場合にも、命令列中で分岐をしなければならない分岐点に充分先立って行なうことができる。そして、コンパイル時に分岐元アドレスと分岐先アドレスを指定する分岐予約命令が分岐点に充分先立って命令列中に配置される。命令フェッチへは通常はプログラムカウンタ（ＰＣ）１２より命令読み込みアドレスがフェッチアドレス１０１のパスへ出力されている。また、一回の命令読み込みごとに次の命令読み込みアドレスがインクリメンタ（“＋１”加算器）１３より計算されて、プログラムカウンタ１２に書き戻される。プログラムカウンタ１２の出力は、また比較器１１にも入力され、分岐アドレススタック１０に登録されている分岐元アドレスと一致していないかどうか、比較されている。図１の例では、分岐アドレススタック１０には複数の分岐元アドレスと分岐先アドレスの組が登録可能であり、その中で最新の組が選ばれて、その分岐元アドレスが比較器１１に入力されている。比較器１１で一致が検出されると、分岐アドレススタック１０に登録されたその組の分岐先アドレスが取り出されセレクタ１４によりフェッチアドレス１０１のパスへ出力されて、命令フェッチに用いられる。該分岐先アドレスが分岐アドレススタック１０から取り出されると、この分岐先アドレスと分岐元アドレスの組が分岐アドレススタック１０から消去され、上の段に格納されていた組が最新の組となる。また同時に、分岐先アドレスがインクリメンタ１３を通してプログラムカウンタ１２に書き戻されるので、分岐先アドレスから以後再び連続して命令を読み込むことが可能となる。

【００１１】図２では、プロセッサ１と主記憶２について示してある。プロセッサ１では、図１のような構成を取る命令フェッチアドレス生成部２１を持つほかに、読み込んだ命令を解読する命令デコード部２２、解読結果に従って命令を実行する命令実行部２３、及びプロセッサ内でデータを蓄えるレジスタファイル２４を備えている。また主記憶２は、プロセッサ１の外の記憶要素であり、プロセッサ１とは、フェッチアドレス１０１、フェ

ッチ命令２０２、データアドレス２０３、書き込みデータ２０４、読み出しデータ２０５の各データパスによって接続されている。主記憶２は、まず、フェッチアドレス１０１のパスに出力された命令アドレスに従って記憶されている命令をフェッチ命令２０２のパスに返す。またデータアドレス２０３に出力されたデータアドレスに従って書き込みデータ２０４のパスに出力されたデータを記録したり、読み出しデータ２０５のパスに記憶されたデータを出力したりする。この主記憶２にはキャッシュを含んでいても良い。主記憶２よりフェッチ命令２０２のパスに読み出された命令は、フェッチアドレス１０１による命令アドレスと共に命令デコード部２２に送られる。命令デコード部２２では、命令を解読して解読結果を命令実行部２３に送ると同時にレジスタファイル２４より必要なレジスタの内容を読み出して、同じく命令実行部２３に送る。命令実行部２３では、送られた命令解読結果及びレジスタの内容を用いて命令を実行し、実行結果をレジスタファイル２４に書き戻したり、主記憶２５に対してデータの読み出し／書き込みを行ったりする。特に分岐予約命令の時には、実行の結果定まる分岐元アドレスと分岐先アドレスの値をそれぞれ分岐元アドレス２０６、分岐先アドレス２０７のパスに出力し、命令フェッチアドレス生成部２１に送る。図２では、以上のような命令フェッチアドレス生成部２１、命令デコード部２２、及び命令実行部２３の動作は並列に行われているものとする。即ち、これによって命令の処理がパイプライン化されている。

【００１２】図３は、命令フェッチアドレス生成部２１についての、本発明の別の実施例である。図３では、分岐アドレスレジスタ３０に一組だけの分岐元アドレス及び分岐先アドレスの値が保持されている。また命令実行部からは、図１の実施例では分岐元アドレスと分岐先アドレスの設定信号だけであったが、図３の例ではそれに加えて設定取り消し信号も備えられている。分岐先アドレスが分岐アドレスレジスタ３０から取り出されると、この分岐先アドレスの組、すなわち分岐先アドレスと分岐元アドレス、が分岐アドレスレジスタ３０から消去されることは図１の場合と同様である。命令実行部において条件取り消し命令が実行され、その条件取り消し命令で指定された条件（例えば、フラグＡ＝１）が成立していると、この設定取り消し信号が出力され、これにより分岐アドレスレジスタ３０に一旦登録されていた分岐元アドレスと分岐先アドレスが取り消されて無効となる。これにより条件分岐の処理を実現している。その他の比較器１１、プログラムカウンタ１２、インクリメンタ１３、セレクタ１４の構成は図１と同じである。

【００１３】図８は、さらに別の実施例であり、図１に示す構成に図３に示す設定取り消し信号を備えたものである。この設定取り消し信号により分岐アドレススタック１０の最下段の組、すなわち最新の予約の組、を取り消すようにしている。なお、分岐アドレススタック１０に設定された複数の予約の組の内どれを取り消すかを指定するようにしてもよい。

【００１４】図４は、分岐予約命令の例および分岐の予約を取り消す命令の例である。図４（ａ）は最も基本的な分岐予約命令の形式であり、分岐元アドレス（ｆｒｏｍ＿ａｄｄｒ）、及び分岐先アドレス（ｔｏ＿ａｄｄｒ）の二つを命令中で値を指定する即値オペランドとして持つ。この二つの即値オペランドの内のどちらか、あるいは両方とも、現在のプログラムカウンタの値からの相対値としても良い。相対値によるアドレス計算が必要であれば、例えば図２の例では命令実行部２３で計算させることができる。

【００１５】図４（ｂ）は他の分岐予約命令の形式であり、図４（ａ）に対し、分岐先アドレスをレジスタ（ｒｅｇ）の内容で指定できるようにした例である。この命令が実行されると、上記レジスタ（ｒｅｇ）の内容を参照して、分岐先アドレスを求め、命令で指定された分岐元アドレスと求められた分岐先アドレスを組として分岐アドレススタックあるいは分岐アドレスレジスタに設定する。このような命令形式は、サブルーチンからの戻り（リターン）など、実行時に動的に分岐先が決まる場合や、図４（ａ）では二つの即値に対して、（コンパイラにより機械語を生成する際の）命令のエンコードに必要なビット数が不足して充分大きな値を設定できない場合にも有効である。

【００１６】図４（ｃ）は、一旦予約した分岐を条件的に取り消す命令の例である。これを図４（ａ）または（ｂ）のような分岐予約命令と組み合わせれば、条件分岐の処理を実現できる。対象とするプロセッサで図８の実施例のように同時に複数の分岐予約ができる場合には、どの予約を取り消すのかも指定する必要があるが、ここでは図８の説明で述べたように、常に最新の予約の組を取り消すと定める。そのため図４（ｃ）では、取り消すための条件のみを命令中のｃｏｎｄｉｔｉｏｎで指定する形式となっている。すなわち、条件として、例えば、フラグＡ＝１を指定すれば、フラグＡ＝１が成立している場合、設定取り消し信号が分岐アドレススタック１０に与えられ、最新の予約の組が強制的に取り消される。このように複数の組から最新のものを選ぶ機能は、図１のような実施例では、比較器１１に供給する分岐元アドレスをどの組から選ぶか決定されている際に用いられるものであるので、これを取り消す組の選択にも利用することが可能である。

【００１７】図４（ｄ）は、一旦予約した分岐を条件的に取り消す命令の別の例である。これは、指定された二つのレジスタ％ａ及び％ｂの内容を比較し、更にｃｏｎｄｉｔｉｏｎで指定した条件が成立していれば、図４（ｃ）と同じく最も最近に予約された分岐を取り消す。この命令は、条件分岐の処理によく現われる比較を行う

命令と、図４（ｃ）にあるような取り消す条件のみを指定する条件付き取り消し命令との融合命令である。このような命令を設けることにより、特に図４（ｃ）のような予約した分岐を取り消すための命令を別個に実行する必要がなくなり、命令数が削減される。また図４（ｄ）では、命令で指定しなければならないオペランドは比較の対象となるレジスタ二つと、分岐を取り消すための条件の計三つであり、簡単であるので、命令のエンコードにも難点はない。もし、命令のエンコードに難点が予想されるような複雑な条件を指定するなど、図４（ｄ）のように一命令で実現するには複雑な実施を行う場合には、図４（ｃ）のような条件のみを指定する分岐取り消し命令と、通常の比較命令などを組み合わせて行えばよい。

【００１８】次に本発明を、従来の分岐方式と比較してみる。図５は、本発明による分岐予約を行うプロセッサと、従来のプロセッサでの条件分岐を行う命令列の例である。まず、図５（ａ）にＣ言語による条件処理の例を掲げる。図中指定された条件（ａ＝＝ｂ）が成立するとｔｈｅｎ部が、不成立の時にはｅｌｓｅ部が実行される。図５（ａ）のような条件分岐の処理を従来の分岐命令を持つプロセッサの命令列で実現した例が図５（ｂ）である。図５（ｂ）では、変数ａ及びｂの値が予めレジスタ％ａ及び％ｂにそれぞれ保持されているものとする。まず比較命令　ｃｏｍｐａｒｅ　％ａ、％ｂ　により変数ａ、ｂの値が等しいかどうか調べられ、等しくない時、条件分岐命令　ｂｒａｎｃｈ　ｎｏｔ＿ｅｑｕａｌ、Ｌ１によりｅｌｓｅ部の処理を行うラベルＬ１以下へ分岐する。変数ａ、　ｂの値が等しい時はラベルＬ１への分岐は行われず、ｔｈｅｎ部の処理を行うラベルＬ０以下が実行される。そしてｔｈｅｎ部の最後に、ラベルＬ１以下のｅｌｓｅ部の処理を行なわずに、処理をラベルＬ２以下の処理に続けるために、無条件分岐命令　ｂｒａｎｃｈ　Ｌ２　によりラベルＬ２へ分岐する。

【００１９】これに対し、本発明による分岐予約方式の命令列は図５（ｃ）のようになる。図５（ｃ）では、初めに条件が不成立だったとき実行されるラベルＬ１以下のｅｌｓｅ部への分岐を命令　ｂｒａｎｃｈ＿ｔａｒｇｅｔ　Ｌ０、Ｌ１　により予約しておく。図５（ｃ）でも図５（ｂ）と同じく、変数ａ及びｂの値は予めレジスタ％ａ及び％ｂにそれぞれ保持されているものとする。これにより、比較付き条件取り消し命令　ｃｏｍｐａｒｅ＿ａｎｄ＿ｐｏｐ　％ａ、％ｂ、ｅｑｕａｌ　でまず変数ａ、ｂの値の比較を行う。そして変数ａ、　ｂの値が等しいという条件が成立していた時のみ、ｅｌｓｅ部への分岐の予約を取り消す。実際には、この予約の取り消しも分岐点Ｌ０の命令の読み込み前に終了できるようにコンパイラで命令列のスケジューリングの最適化が行われるものとする。もし最適化が行われず条件取り消し命令による予約の取り消しが分岐点の命令の実行の直前

となり、その読み込みに間に合わなかったときには、予約に従って読み込まれていた命令を取り消すなどの制御が必要である。逆に図５（ｂ）のような従来のプロセッサの命令列との大きな違いは、分岐点Ｌ０までの命令がすべて予約に関する命令であり、分岐点Ｌ０（の直前）には何の命令も置く必要はないので、分岐点Ｌ０よりかなり以前に遡る命令のスケジューリングも可能であると言うことである。即ちコンパイラにおける命令スケジューリングの最適化に関する自由度が大きい。図５（ｃ）において、条件取り消し命令によって分岐予約が取り消された時にはラベルＬ０以下のｔｈｅｎ部が実行されるが、その先頭で命令　ｂｒａｎｃｈ＿ｔａｒｇｅｔ　Ｌ１、　Ｌ２　でｔｈｅｎ部の終り（ｅｌｓｅ部の直前）Ｌ１からの分岐を予約し、ｔｈｅｎ部の処理の終了後、ラベルＬ１以下のｅｌｓｅ部の処理をすることなく、滞り無く次のラベルＬ２以下の処理を続けられるようにしている。この場合も分岐点となるラベルＬ１（の直前）には何の命令も置く必要もなく、命令のスケジューリングに大きな自由度がある。

【００２０】図６は、本発明による分岐予約を行うプロセッサでの、サブルーチン呼び出し／戻りを行う命令列の例である。サブルーチン呼び出しでは、サブルーチンの先頭アドレスは多くの場合静的に決定でき、本発明の分岐予約方式を容易に適用できる。またサブルーチンからの戻り（リターン）は、戻り先が一般には動的に決定されるが、その決定の時点はサブルーチンの呼び出し時であり、実際の分岐点であるサブルーチンの終りに充分先立って決定されている。これらのことから、サブルーチンの呼び出し／戻りにも、本発明による分岐予約方式を適用して効率を高めることができる。

【００２１】図６において、サブルーチンＬ２の呼び出しを命令　ｂｒａｎｃｈ＿ｔａｒｇｅｔ　Ｌ１、Ｌ２　によって実際の呼び出し点Ｌ１に充分先立って予約している。またサブルーチンからの戻りアドレスを設定するために、呼び出し点Ｌ１までに命令　ｍｏｖ　Ｌ１、％ｒ１　で戻りアドレスＬ１（図６の例では呼び出し点に同じ）をレジスタ％ｒ１に設定しておく。呼び出された側のサブルーチンＬ２では、サブルーチンの終りＬ３までにレジスタ％ｒ１の内容を分岐先とする分岐予約命令　ｂｒａｎｃｈ＿ｏｎ＿ｒｅｇｉｓｔｅｒ　Ｌ３、％ｒ１　により戻り先を予約しておく。以上のような命令列によって、呼び出し点Ｌ１よりサブルーチンＬ２が呼び出され、サブルーチンの終りＬ３から再び戻り点Ｌ１へ戻るという動作が分岐予約により実現される。

【００２２】次に実際のパイプラインの動作として、本発明による分岐予約と従来の例を比較してみる。図７は、図１、図３あるいは図８で示したような本発明による分岐予約を行うプロセッサで実際の分岐が生じている時のパイプライン動作と、従来のプロセッサの分岐のパイプライン動作の比較である。ここで、Ｆは命令フェッ

チ、Ｄは命令デコード、Ｅは命令実行、Ｍはメモリ・アクセス、Ｗはレジスタ・ライトバック、また、矢印は分岐アドレスの供給を示す。

【００２３】図７（ａ）は従来の分岐方式による分岐処理時のパイプライン動作の例である。分岐命令の直前（分岐点）の命令、分岐命令、分岐命令の結果次に分岐先の命令を読み込めるまでのディレイスロット１サイクルと、分岐先の命令の各実行サイクルをパイプラインに沿って図示してある。図７（ａ）のように、従来の方式では、最低限分岐命令が読み込まれて命令デコードサイクル（Ｄ）まで達しないと分岐先が決定できない。このため図７（ａ）では、分岐の直前（分岐点）の命令から分岐先の命令までの間に分岐命令とそのディレイスロットの２サイクルが存在する。これに対し、図７（ｂ）に本発明による分岐予約方式のパイプライン動作の例を示す。図７（ｂ）では、分岐点には、分岐のためには何ら命令を必要としない。分岐点の命令（分岐の直前の命令）のフェッチサイクル（Ｆ）において、命令の読み込みが分岐点まで達っしたことが検出されると、図１、図３、図８で説明したような機構で速やかに分岐先の命令の読み込みを開始できる。分岐点の命令と分岐先の命令の間にも余計なサイクルは不要である。このようにパイプラインにおける動作を非常に効率化できる。

【００２４】図９Ａ～Ｃは、本発明の予約による分岐のパイプラインの動作と、従来の予測による分岐のパイプライン動作の違いを示した例である。図９Ａ～Ｃにおいて、Ｆは命令のフェッチ（読み込み）、Ｄは命令のデコード（解読）、Ｅは命令の実行、Ｍはメモリ（キャッシュまたは主記憶）のアクセス、Ｗは命令の結果のレジスタへの書き戻しの動作を行うパイプライン・ステージを示す。また、分岐先アドレスの供給を実線の矢印で表している。更に、分岐点の命令フェッチの開始時点をＳＰ１、分岐先の命令フェッチの開始時点をＳＰ２として示してある。

【００２５】図９Ａは、従来の予測による分岐の方法において、分岐が生じたときに予測が誤っていた場合のパイプライン動作の例である。図９Ａにおいて、命令１は分岐点（分岐命令の直前）の命令、命令２は分岐命令、命令３は分岐命令に後続する、誤った予測によってフェッチされた命令である。また命令４は正しい分岐先の命令である。分岐の予測の誤りは分岐命令（命令２）がＤステージで解読されて、検出可能となる。このとき、分岐命令に後続する命令３は既にＦステージで読み込まれているので、命令３のデコードおよびそれ以降の動作を取り消さなければならない。また同時に正しい分岐先の命令４をフェッチし直す必要がある。正しい分岐先、即ち命令４のアドレスは、分岐命令のＤステージ以降で生成されるので、結果として、分岐先の命令フェッチ（ＳＰ２）は、分岐点の命令フェッチ（ＳＰ１）から３サイクルかかることになる。

【００２６】図９Ｂは、従来の予測による分岐の方法において、分岐が生じたときに予測が正しかった場合である。図９Ｂにおいて、命令１は分岐点（分岐命令の直前）の命令、命令２は分岐命令、命令３は予測によってフェッチされた分岐先の命令である。分岐の予測は分岐命令（命令２）のフェッチによって行われるので、図のように、分岐命令の読み込みに続いて、遅滞なくすぐ次のサイクルで分岐先の命令３のフェッチを開始することが可能である。結果として分岐点の命令フェッチ（ＳＰ１）から２サイクルで分岐先の命令フェッチ（ＳＰ２）が始まっている。

【００２７】図９Ｂのように正しい予測が行われるためには、分岐命令２が１回以上実行されて、分岐先が記録されている必要がある。また同時に予測に用いているアルゴリズムが該分岐命令の動作を正しく予測できなければならない。従って、分岐命令２の初回の実行、および予測のアルゴリズムが正しく予測できないときには、パイプラインの動作は図９Ａに示したようになり、３サイクルかかることになる。

【００２８】これらに対し、図９Ｃに本発明の予約による分岐のパイプライン動作を示す。図９Ｃにおいて、命令１は分岐予約命令、命令２は通常の命令、命令３は分岐点の命令、命令４は分岐先の命令である。パイプラインの動作では、まず、分岐予約命令（命令１）の実行によりＥステージで、分岐点のアドレスと分岐先のアドレスの両方が登録されて、分岐の予約が行われる。分岐予約命令は、命令のフェッチが分岐が生じる分岐点に達する充分前に行われるように、コンパイラによってプログラムのコード生成時に配置される。その後、図９Ｃでは、一個以上の通常命令（命令２）がフェッチされ、実行される。そして、命令のフェッチが分岐点の命令３に達したことがＦステージで検出されると、命令のフェッチアドレスがすぐに分岐予約命令１で登録された分岐先アドレスに切り替えられ、遅滞なく分岐先の命令４のフェッチが行われる。このとき、分岐が生じる分岐点においては、分岐に関る命令はなんら実行される必要はない。分岐点の命令３も命令２と同じく、任意の通常命令でよい。結果として、分岐点の命令３のフェッチ（ＳＰ１）から１サイクルで、分岐先の命令４のフェッチが行われている。図９Ｃのようなパイプライン動作は、本発明の予約による分岐では、分岐予約命令の初回の実行から常に可能である。

【００２９】図１０Ａ～Ｂは、本発明による分岐予約命令を用いてプログラムをコンパイルしてコード生成を行う例を示したものである。図１０Ａでは、プログラムをコンパイルする際の、コンパイラの全体の操作の例を示してある。まずステップ１１０でコンパイルするプログラムのソースが読み込まれ、ステップ１２０で構文解析及び文法チェックが行われる。次にステップ１３０ではプログラムの意味解析が行われ、ステップ１４０で制御

フローグラフを、ステップ１５０でデータフローグラフが生成される。これらのフローグラフの情報を元にステップ１６０で大域的最適化及び局所的最適化が行われ、ステップ１７０で実際の機械コードが生成されて、出力される。

【００３０】図１０Ｂは、図１０Ａのステップ１７０における機械コード生成を詳細に示したものである。図１０Ｂにおいて、まずステップ１７１で各基本ブロック毎にその中に含まれる演算操作が決定される。次にステップ１７２でそれぞれの演算操作に対応する機械命令を生成する。ステップ１７３で機械命令コードにおける最適化を行う。そしてステップ１７４で分岐予約命令を各基本ブロックの先頭に挿入する。最後にステップ１７５で生成した機械コードの列を出力する。

【００３１】図１０Ｃは実際にコンパイルされるプログラムのソースの構造の例を示している。また図１０Ｄは図１０Ｃのプログラムをコンパイルした結果を示している。これらにおいて、基本ブロックはＡ～Ｄの４個が示されている。図１０Ｃでは、プログラムの構造を基本ブロックＡ～Ｄの接続として示してある。即ちプログラムは基本ブロックＡから開始され、基本ブロックＡの終わりから基本ブロックＢまたはＣへ条件分岐を行う。基本ブロックＢ及びＣは基本ブロックＤへ無条件に分岐する。基本ブロックＤをすべて実行するとプログラムが終了する。

【００３２】このような分岐を本発明の分岐予約命令を用いて実現すると、図１０Ｄに示したような機械コード列となる。即ち、基本ブロックＡの先頭には、Ａの終わりからＣの先頭への分岐を予約する分岐予約命令が置かれる。その後、基本ブロックＡの演算を行う機械命令と共に、条件分岐を行うための条件生成の命令が置かれ、これを受けて分岐予約の取り消しを行う条件取消命令も置かれる。基本ブロックＡの命令列の直後には基本ブロックＢの命令列が配置され、基本ブロックＡの命令列において生成された条件が成立していると、条件取消命令により、基本ブロックＡの先頭で予約された分岐が取り消され、命令の実行はそのまま基本ブロックＢに移る。このとき、条件取消命令で指定する条件は分岐を取り消す条件、即ち分岐しない条件である。この条件が不成立の時には、条件取消命令が実行されても予約の取り消しは行われず、基本ブロックＡの命令列の最後から（最後の命令をフェッチすると）、予約された分岐が実行され、命令の実行が基本ブロックＣに移る。基本ブロックＢの先頭には、Ｂの終わりからＤの先頭への分岐を予約する分岐予約命令が置かれ、Ｂの終わりからは（無条件に）Ｄの先頭への分岐が起こる。従って、図１０Ｄのように基本ブロックＣの命令列を基本ブロックＢの命令列の直後に配置することが可能であり、基本ブロックＤの命令列をＣの直後に配置しておけば、図１０Ｃに示した基本ブロックＢ及びＣから基本ブロックＤへの無条件分岐が表現できる。

【００３３】図１１Ａ～Ｅは、本発明の予約による分岐について、更に詳細な動作の例を説明したものである。図１１Ａにおいて、メモリ内にはアドレス９９に、分岐点アドレスが１０２、分岐先アドレスが１０００の分岐予約命令が置かれている。プロセッサ内において、図１１Ａでは、このアドレス９９の分岐予約命令の読み込みは終了しており、命令実行部において実行が開始されている。命令実行部における実行の結果、分岐アドレススタックには、分岐点アドレス１０２と分岐先アドレス１０００のアドレスペアが登録され、分岐が予約される。その後、図１１Ｂにおいてプログラムカウンタ（ＰＣ）により保持されている命令フェッチアドレスが分岐アドレススタックに登録された分岐点アドレス１０２と等しくなると、図１１Ｃにおいて命令フェッチアドレスが登録された分岐先１０００に切り替えられる。図１１Ｃで命令フェッチアドレスの切り替えが行われている間に分岐点の命令１０２の読み込みが完了し、図１１Ｄで分岐先１０００の命令の読み込みが行われている間に並列に実行されている。同様に、分岐先命令の読み込みは図１１Ｄで完了し、図１１Ｅで後続のアドレス１００１の命令が読み込まれている間に、これと並列に実行される。

【００３４】図１２Ａ～Ｃは、従来の予測による条件分岐のパイプライン動作の様子と、本発明の分岐予約による条件分岐のパイプライン動作の様子の違いの例を示したものである。図１２Ａ～Ｃにおいて、Ｆは命令のフェッチ（読み込み）、Ｄは命令のデコード（解読）、Ｅは命令の実行、Ｍはメモリ（キャッシュまたは主記憶）のアクセス、Ｗは命令の結果のレジスタへの書き戻しの動作を行うパイプライン・ステージを示す。また、図１２Ａ、Ｂでは、分岐先アドレスの供給を実線の矢印で、分岐点の命令フェッチの開始時点をＳＰ１、分岐先の命令フェッチの開始時点をＳＰ２として示してある。図１２Ｃでは、条件分岐に関る条件取消命令のフェッチ開始時点をＳＰ１とし、分岐後の命令フェッチの開始時点をＳＰ２としている。

【００３５】図１２Ａは、従来の予測による条件分岐で、予測が誤っていた場合の例である。図１２Ａで、命令１は比較命令、命令２は条件分岐命令、命令３は誤った予測によってフェッチされた命令、命令４は正しい分岐先の命令である。命令３のＤステージ以降の処理は、条件分岐命令（命令２）がＤステージで解読されて予測が誤っていたことが検出されると、取り消される。その際、正しい分岐先のアドレスが生成されて、Ｆステージに伝達され、命令４のフェッチが行われる。この結果、分岐点、即ち条件分岐命令（命令２）の直前の比較命令１のフェッチ時点ＳＰ１から、正しい分岐先の命令４のフェッチ時点ＳＰ２まで３サイクルかかる。

【００３６】図１２Ｂは、従来の予測による条件分岐で、予測が正しかった場合のパイプライン動作の例であ

る。図１２Ｂで、命令１は比較命令、命令２は、条件分岐命令、命令３は予測による（この場合は正しい）分岐先の命令である。図のように、条件分岐命令（命令２）のフェッチで分岐先のアドレスが予測されるので、予測が正しい場合には、すぐ次のサイクルで分岐先の命令４のフェッチを行うことができる。結果として、分岐点の命令１のフェッチＳＰ１から分岐先の命令４のフェッチまで２サイクルになる。このように分岐先のアドレスまで正しく予測できるためには、図９Ｂとの場合と同じく、条件分岐命令２が一度以上実行されて、分岐先のアドレス計算が行われ、何等かの分岐予測の手段により記録されていなければならない。また同時に予測に用いるアルゴリズムが条件分岐の成立／不成立を正しく予測できなければならない。

【００３７】図１２Ｃは、本発明の分岐予約による条件分岐のパイプライン動作の例を示したものである。特に、分岐が生じる条件が成立せず、分岐しなかった場合を示す。図１２Ｃで、命令１は分岐予約命令、命令２は条件取消命令、命令３は通常の命令、命令４は分岐点にあたる通常の命令、命令５は分岐点に連続する（分岐しなかったときの）命令である。これらの内、命令３、４は任意の有効な通常命令を充てることが出来る。図１２Ｃでは、分岐予約命令（命令１）の実行によりＥステージで分岐点と分岐先のアドレスが予約される。分岐先アドレスは、図の点線の矢印で示すように、分岐点に後続する命令５のＦステージに伝達されようとする。しかし条件取消命令（命令２）の実行によりＥステージで取り消しが生じて（図中の“Ｘ”印）、分岐が生じず、命令５は分岐点の命令４に後続する命令となる。以上の動作により、図１２Ｃでは、分岐点の命令４のフェッチから遅滞なく１サイクルで後続する命令５のフェッチを行うことが出来る。条件分岐に関る条件取消命令（命令２）のフェッチ時点（ＳＰ１）から数えても、３サイクルで条件分岐を行うことが出来る。

【００３８】このように、本発明の予約により分岐する方法においても、条件分岐を効率良く行うことができる。

【００３９】

【発明の効果】本発明により、命令単位の並列実行を行うプロセッサを含むパイプライン化されたプロセッサにおいて、命令の実行と命令の読み込みの並列動作をなんら妨げることなく、効率的に分岐動作を行わせることができる。

【図面の簡単な説明】

【図１】本発明によるプロセッサの命令フェッチアドレス生成部の一実施例を示す図である。

【図２】プロセッサの全体構成の例を示す図である。

【図３】本発明によるプロセッサの命令フェッチアドレス生成部の第２の実施例を示す図である。

【図４】本発明による分岐予約を行うための分岐予約命令の例および分岐予約の取り消し命令を示す図である。

【図５】本発明による条件分岐処理と従来技術による条件分岐処理の比較説明のための命令列の例を示す図である。

【図６】本発明による分岐予約命令を用いたサブルーチン呼び出し及び戻りの例を示す図である。

【図７】本発明による分岐予約命令と従来の分岐命令のパイプライン動作の比較を説明するための図である。

【図８】本発明によるプロセッサの命令フェッチアドレス生成部の第３の実施例を示す図である。

【図９Ａ】従来の予測において、分岐が生じたときに予測が誤っていた場合のパイプライン動作の例を示す図である。

【図９Ｂ】従来の予測において、分岐が生じたときに予測が正しかった場合のパイプライン動作の例を示す図である。

【図９Ｃ】本発明の予約による分岐のパイプライン動作の例を示す図である。

【図１０Ａ】プログラムをコンパイルする際の、コンパイラの全体の操作の例を示す図である。

【図１０Ｂ】図１０Ａのステップ１７０における機械コード生成を詳細に示す図である。

【図１０Ｃ】実際にコンパイルされるプログラムのソースの構造の例を示す図である。

【図１０Ｄ】図１０Ｃのプログラムをコンパイルした結果を示す図である。

【図１１Ａ】本発明の予約による分岐動作の例を説明するための図である。

【図１１Ｂ】図１１Ａの分岐動作の続きを説明するための図である。

【図１１Ｃ】図１１Ｂの分岐動作の続きを説明するための図である。

【図１１Ｄ】図１１Ｃの分岐動作の続きを説明するための図である。

【図１１Ｅ】図１１Ｄの分岐動作の続きを説明するための図である。

【図１２Ａ】従来の予測による条件分岐で、予測が誤っていた場合のパイプライン動作の例を示す図である。

【図１２Ｂ】従来の予測による条件分岐で、予測が正しかった場合のパイプライン動作の例を示す図である。

【図１２Ｃ】本発明の分岐予約による条件分岐のパイプライン動作の例を示す図である。

【符号の説明】

１　プロセッサ
２　主記憶
１０　分岐アドレススタック
１１　比較器
１２　プログラムカウンタ
１３　インクリメンタ
１４　セレクタ

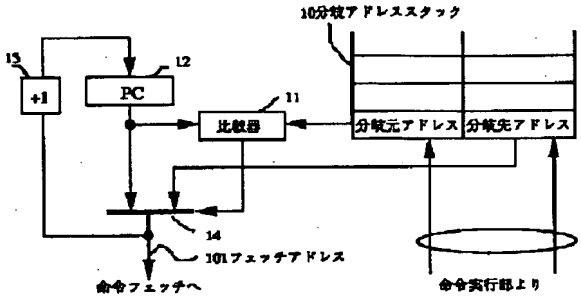
17

２１　命令フェッチアドレス生成部
２２　命令デコード部
２３　命令実行部

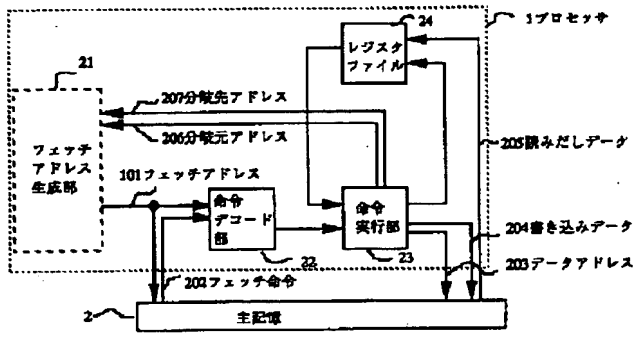
18

２４　レジスタファイル
３０　分岐アドレスレジスタ
１０１　フェッチアドレス

【図１】



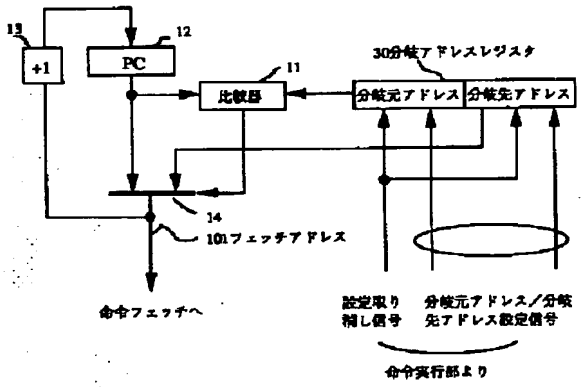
【図２】



【図３】

【図４】

【図３】



【図４】

(a)	branch_target	from_addr, to_addr
(b)	branch_on_register	from_addr, reg
(c)	pop_target	condition
(d)	compare_and_pop	%a, %b, condition

【図７】

【図５】

【図５】

(a)

```
if (a == b)
        … (then部)
else
        … (else部)
```

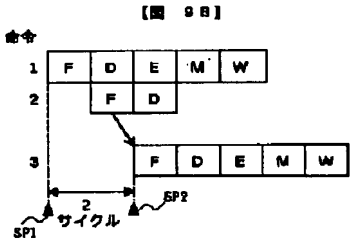
(b)

```
        compare         %a, %b
        branch          not_eqaul, L1
L0:
        … (then部)
        branch          L2
L1:
        … (else部)
L2:
        …
```

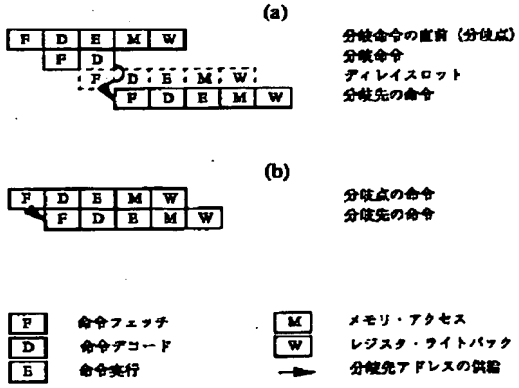
(c)

```
        branch_target   L0, L1
        compare_and_pop %a, %b, equal
L0:
        branch_target   L1, L2
        … (then部)
L1:
        … (else部)
L2:
        …
```

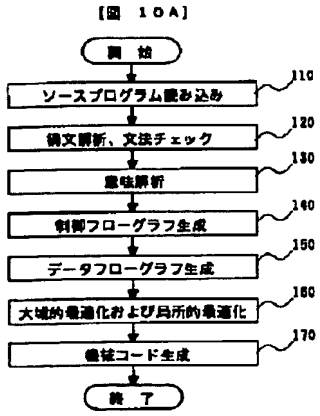
【図９Ｂ】



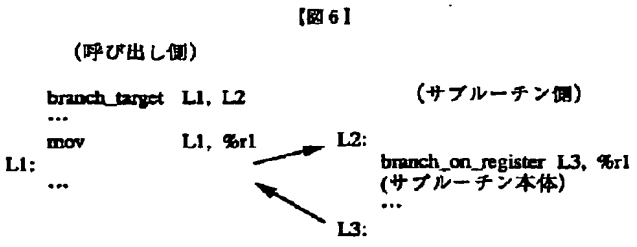
【図７】



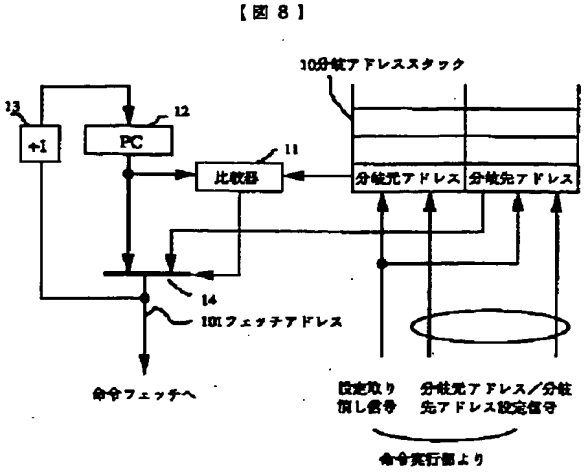
【図１０Ａ】



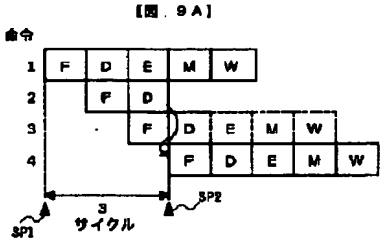
【図６】



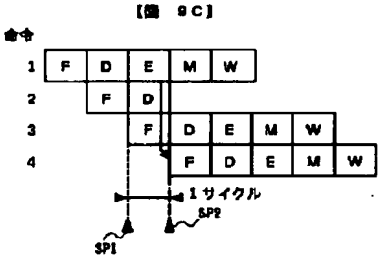
【図８】



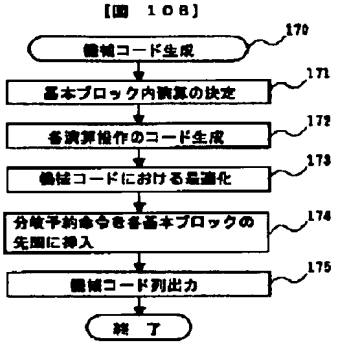
【図９Ａ】



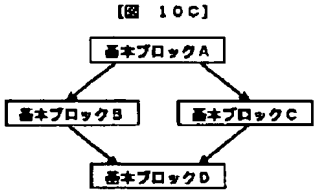
【図９Ｃ】



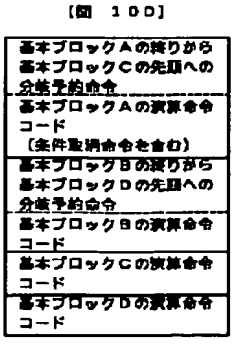
【図１０Ｂ】



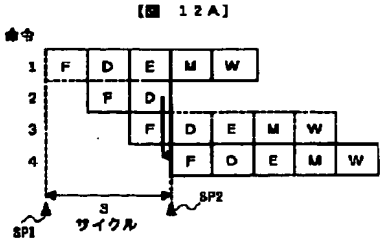
【図１０Ｃ】



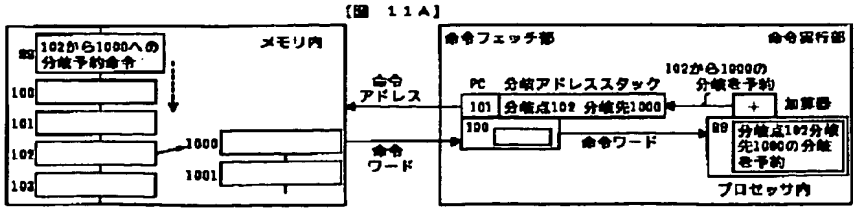
【図１０Ｄ】



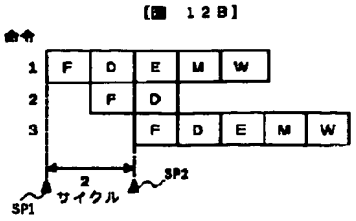
【図１２Ａ】



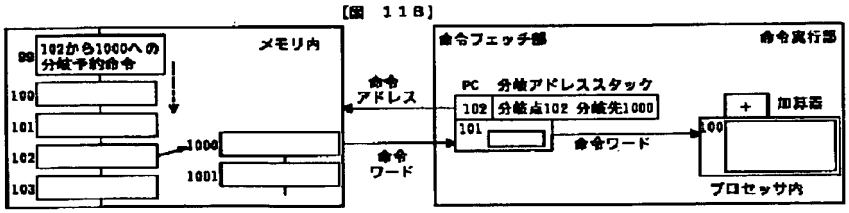
【図１１Ａ】



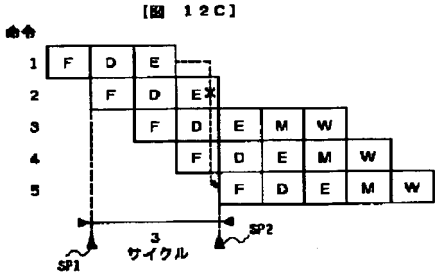
【図１２Ｂ】



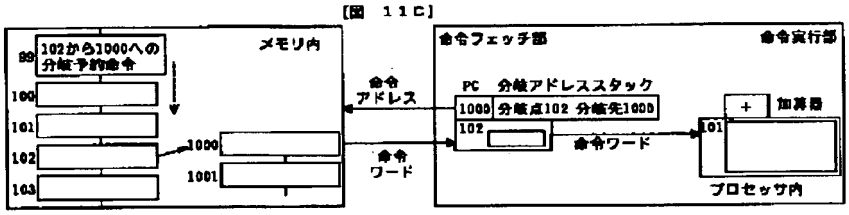
【図１１Ｂ】



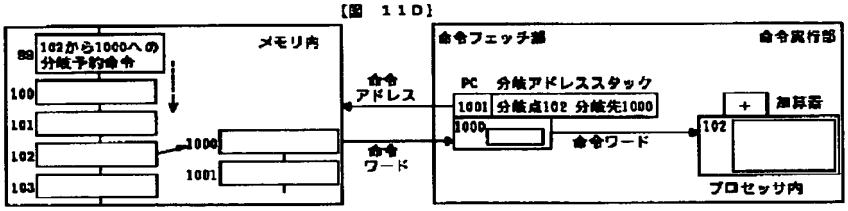
【図１２Ｃ】



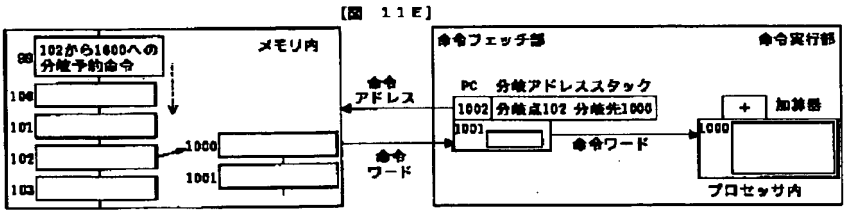
【図１１Ｃ】



【図１１Ｄ】



【図１１Ｅ】



フロントページの続き

(72)発明者　金子　憲二
東京都国分寺市東恋ケ窪１丁目280番地
株式会社日立製作所中央研究所内